ArcPy.Mapping Module Makes Complex PDF Creation Easy

Contributed by Scott Davis
16, Mar. 2011
Last Updated 16, Mar. 2011

Recently, I was presented with a problem that was a perfect opportunity for trying out ESRI's ArcPy Python site package. We have a series of map documents that are set up with Data Driven Pages to export various maps for each of the counties of Utah. Each mxd had a different theme. The goal was to develop a script that would export all of the Data Driven Pages for each mxd and then combine them by county. After a few hours of work I had 72 lines of code that did just that. Here's what I came up with:

I used only two modules for this script: arcpy.mapping and os (great for working with the file system).

```
# import modules
import arcpy.mapping, os

# variables
baseFolder = os.getcwd() # current working directory
outputFolder = baseFolder + r'\PDFs'
```

bt_code_init('995b4ad7-40a6-4fb3-8615-c1f5c5b87e0d');

The os module was great for deleting the old files and getting a list of the map documents.

```
# clear out old pdfs
print '\nDeleting old PDFs...'
oldPDFs = os.listdir(outputFolder)
for f in oldPDFs:
    os.remove(outputFolder + '\\' + f)

# get list of all files in the folder
print '\nGetting list of mxds...'
allItems = os.listdir(baseFolder)
# filter out just .mxd's
mxdFileNames = [(x) for x in allItems if x.endswith('.mxd')]
mxdFileNames.sort()
```

bt_code_init('83f9eca3-365e-49c9-ae4b-2a92e58d3415');

The DataDrivenPages class was the key class in the ArcPy.Mapping module for this script. It is obtained through the MapDocument class. Here I start to loop through the mxd's and get a reference to the DataDrivenPages object that I am interested in.

```
# loop through mxds
```

```
for name in mxdFileNames:
    print '\nProcessing: ' + name

    # get mxd
    mxd = arcpy.mapping.MapDocument(baseFolder + '\\' + name)

    # get datadrivenpages object
    ddp = mxd.dataDrivenPages
```

bt_code_init('ae0e8d87-0bc8-4f7d-bbb0-f05fec019cd6');

Once I've got the DataDrivenPages object, then I start to loop through all of the pages.

```
    # loop through pages
    pg = 1
    while pg <= ddp.pageCount:
        # change current page
        ddp.currentPageID = pg

        # get name of current page
        name = ddp.pageRow.getValue('NAME')
        print name
```

bt_code_init('7a24ba44-9e41-401c-a3bb-28fe650a7683');

Before I export the page, I check to see if there is already an existing pdf for that particular county. If there is I export the page to a temp PDF file and then use the PDFDocument::appendPages() function to add it to the existing PDF. If not, then I just export it out to a new PDF.

```
        # check to see if there is already a pdf file created for this county
        pdfFile = outputFolder + '\\' + name + '.pdf'
        if os.path.exists(pdfFile):
            print 'Existing pdf found. Appending...'

            # open PDF document
            pdf = arcpy.mapping.PDFDocumentOpen(pdfFile)

            # output to temporary file
            tempFile = outputFolder + '\\temp.pdf'
            ddp.exportToPDF(tempFile, 'CURRENT')

            # append to existing file
            pdf.appendPages(tempFile)

            # delete temp file
            os.remove(tempFile)

            # clean up variables
            del pdf
```

```
        else: # file does not exist, export to new file
            print 'No existing pdf found. Exporting to new pdf.'
            ddp.exportToPDF(pdfFile, 'CURRENT')

        # increment page number
        pg = pg + 1
```

bt_code_init('4053d304-81d9-42c3-a5b9-37310468e5a6');

Then, all that's left if a little clean up.

```
    # clean up variables
    del mxd, ddp

raw_input('Done. Press any key to exit...')
```

bt_code_init('f3fb34a1-443a-4248-a8c8-a3d49d5a660a');

And that's it! Here's the entire script and an example output pdf.